IN THE SPECIFICATION:

Please amend the specification as follows:

Pursuant to 37 CFR § 1.121(b)(1)(iii), a marked up copy of each paragraph amended below appears on the page immediately following each amendment.

Please delete the paragraph that begins on page 4, line 21 and ends on page 5, line 6 and insert the following paragraph therefor:

--In the process of manufacturing computer system 100, a computer manufacturer stores a plurality of test modules 106 onto computer system 100 as indicated by arrow 108. As noted above, the term test module refers to a set of information that may be used to test or perform diagnostics or other services on a component of a computer system. The information may include, for example, one or more test routines, test information, device information, parameter information, and other programs. The test modules may be used to perform tests on the components of a computer system. Test modules 106 are configured to operate in a diagnostic architecture described herein. In particular, each test module 106 is configured to expose a test module interface. A test module interface provides a user interface, an executive, or other program with the ability to access information and call functions within the test module. In the following description, an executive will be described as interacting with a test module interface. It is noted, however, that a user interface or other program may also interact with a test module interface in ways similar to those described below--

Please delete the paragraph that begins on page 9, line [illegible] and ends on page 9, line 23 and insert the following paragraph therefor:

--Fig. 2 is a diagram illustrating layers of software in a diagnostic architecture. Fig. 2 shows a high level diagram of the diagnostic architecture described in Fig. [illegible]re. Fig. 2, a user interface layer 210, an executive module layer 220, a test module layer 230, and a driver layer 240 are shown. Each software layer 210, 220, 230, and 240 allows other layers to interact with it using an interface such as the test module interface described above.--

Please delete the paragraph that begins on page 9, line 24 and ends on page 10, line 4 and insert the following paragraph therefor:

--Test modules and device drivers are loaded into a memory and expose their interfaces. In some instances, a test module is included in a device driver. The test module interface provides a way for an executive to access information in the test module. Similarly, an executive exposes an executive module interface to allow a test module and a user interface to access information in the executive module, and the user interface exposes a user interface interface to allow an executive module to access information in the user interface. Each interface may utilize an EFI protocol as noted above. The executive module and/or the user interface may cause the test modules to be loaded into or unloaded from a memory.--

Please delete the paragraph that begins on page 10, line 15 and ends on page 10, line 22 and insert the following paragraph therefor:

--The arrows in Fig. 3 represent pointers to interfaces that are stored by each program. For example, user interface 310 stores a pointer 314 to EMIF 324, and

executive module 320 stores a pointer 326 to UIIF 326. Executive module 320 detects the test module interfaces using EFI and stores pointers 328a, 328b, and 328c to TMIFs 330a, 330b, and 330c, respectively. Executive module 320 also detects TMIF 327 and interface 342 of registration module 340 and stores pointers 327 and 329, respectively. As described above, test modules 330a, 330b, and 330c may cause their respective TMIFs 332a, 332b, and 332c to be installed in response to the presence or absence of certain conditions.--

Please delete the paragraph that begins on page 11, line 12 and ends on page 12, line 2 and insert the following paragraph therefor:

--The operation of registration table 400 is seen by way of an example. Figs. 4b, 4c, and 4d are diagrams illustrating registration table 400 as it might appear at various times in the example of Fig. 3. In response to detecting TMIFs 332a, 332b, and 332c, executive module 320 registers its use of test modules 330a, 330b, and 330c, respectively, using registration module 340. Registration module 340 creates an entry for each of these uses such as entries 422, 424, and 426 in registration table 400 shown in Fig. 4b. As indicated in Fig. 4b, entry 422 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332a), and a function indicator (FQUITUSING). Similarly, entry 424 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332b), and a function indicator (FQUITUSING), and entry 426 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332c), and a function indicator (FQUITUSING). Executive module 320 ID may be executive module 320's globally unique identifier ("GUID") or another unique number assigned by EFI in each entry 422, 424, and 426. The pointers in each entry may be pointers to TMIF 332a, TMIF 332b, and TMIF 332c, respectively. The functions in each entry may refer to a function in executive module 320 that may be called to request that executive module 320 stop

using the test modules 330a, 330b, and 330c associated with TMIF 332a pointer, TMIF 332b pointer, and TMIF 332c pointer, respectively.--

Please delete the paragraph that begins on page 13, line 3 and ends on page 13, line 15 and insert the following paragraph therefor:

--At some point in operation, test module 330c or another test module may attempt to reinstall TMIF 332c as indicated by dashed arrow 350. Test module 330c and registration module 340 may follow an analogous notification process just described before test module 330c reinstalls TMIF 332c. After executive module 320, test module 330a, and test module 330b respond to registration module 340 with a signal indicating that they have stopped using test module 330c, registration module 340 notifies test module 330c that it may reinstall TMIF 332c. Registration module 340 also deletes the entries in registration table 400 associated with test module 330c to result in registration table 400 shown in Fig. 4d. Test module 330c can then delete and reinstall TMIF 332c. If either executive module 320, test module 330a, or test module 330b respond to registration module 340 with a signal indicating that test module 330c is still being used, then registration module 340 notifies test module 330c that it may not reinstall TMIF 332c. Test module 330c can then cancel the reinstall and may attempt to reinstall TMIF 332c at a later time.--

MARKED UP COPY OF AMENDMENT PURSUANT TO 37 CFR § 1.121 (b)(1)(iii)

Page 4, line 21 to page 5, line 6.

-- In the process of manufacturing computer system 100, a computer manufacturer stores a plurality of test modules 106 onto computer system 100 as indicated by arrow 108. As noted above, the term test module refers to a set of information that may be used to test or perform diagnostics or other services on a component of a computer system. The information may include, for example, one or more test routines, test information, device information, parameter information, and other programs. The test modules may be used to perform tests on the components of a computer system. Test modules 106 are configured to operate in a diagnostic architecture described herein. In particular, each test module 106 is configured to expose a test module interface. A test module interface provides a user interface, an executive, or other program with the ability to access information and call functions within the test module. In the following description, an executive will be described as interacting with a test module interface. It is noted, however, that a user interface or other program may also interact with a test module interface in ways similar to those described below.--

Page 9, line 19 to page 9, line 23.

-- Fig. 2 is a diagram illustrating layers of software in a diagnostic architecture. Fig. 2 shows a high level diagram of the diagnostic architecture described in Fig. 1. In Fig. 2, a user interface layer 210, an executive module layer 220, a test module layer 230, and a driver layer 240 are shown. Each software layer 210, 220, 230, and 240 allows other layers to [interacts] interact with it using an interface such as the test

module interface described above.--

Page 9, line 24 to page 10, line 4.

-- Test modules and device drivers are loaded into a memory and expose their interfaces. In some instances, a test module is included in a device driver. The test module interface provides a way for an executive to access information in the test module. Similarly, an executive exposes an executive module interface to allow a test module and a user interface to access information in the executive module, and the user interface exposes a user interface interface to allow an executive module to access information in the user interface. Each interface may utilize an EFI protocol as noted above. The executive module and/or the user interface may cause the test modules to be loaded into or unloaded from a memory. --

Page 10, line 15 to page 10, line 22.

-- The arrows in Fig. 3 represent pointers to interfaces that are stored by each program [to interfaces]. For example, user interface 310 stores a pointer 314 to EMIF 324, and executive module 320 stores a pointer 326 to UIIF 326. Executive module 320 detects the test module interfaces using EFI and stores pointers 328a, 328b, and 328c to TMIFs 330a, 330b, and 330c, respectively. Executive module 320 also detects TMIF 327 and interface 342 of registration module 340 and stores pointers 327 and 329, respectively. As described above, test modules 330a, 330b, and 330c may cause their respective TMIFs 332a, 332b, and 332c to be installed in response to the presence or absence of certain conditions.--

Page 11, line 12 to page 12, line 2.

-- The operation of registration table 400 is seen by way of an example. Figs. 4b, 4c, and 4d are diagrams illustrating registration table 400 as it might appear [a] at various times in the example of Fig. 3. In response to detecting TMIFs 332a, 332b, and 332c, executive module 320 registers its use of test modules 330a, 330b, and 330c, respectively, using registration module 340. Registration module 340 creates an entry for each of these uses[—] such as entries 422, 424, and 426 in registration table 400 shown in Fig. 4b. As indicated in Fig. 4b, entry 422 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332a), and a function indicator (FQUITUSING). Similarly, entry 424 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332b), and a function indicator (FQUITUSING), and entry 426 includes a module ID (executive module 320 ID), a pointer (TMIF pointer 332c), and a function indicator (FQUITUSING). Executive module 320 ID may be executive module 320's globally unique identifier ("GUID") or another unique number assigned by EFI in each entry 422, 424, and 426. The pointers in each entry may be pointers to TMIF 332a, TMIF 332b, and TMIF 332c, respectively. The functions in each entry may refer to a function in executive module 320 that may be called to request that executive module 320 stop using the test modules 330a, 330b, and 330c associated with TMIF 332a pointer, TMIF 332b pointer, and TMIF 332c pointer, respectively. --

Page 13, line 3 to page 13, line 15.

-- At some point in operation, test module 330c or another test module may attempt to reinstall TMIF 332c as indicated by dashed arrow 350. Test module 330c and registration module 340 may follow an analogous notification process just described before test module 330c reinstalls TMIF 332c. After executive module 320, test module 330a, and test module 330b respond to registration module 340 with a signal indicating

that they have stopped using test module 330c, registration module 340 notifies test module 330c that it may reinstall TMIF 332c. Registration module 340 also deletes the entries in registration table 400 associated with test module 330c to result in registration table 400 shown in Fig. 4d. Test module 330c can then delete and reinstall TMIF 332c. If either executive module 320, test module 330a, or test module 330b respond to registration module 340 with a signal indicating that test module 330c is still being used, then registration module 340 notifies test module 330c that it may not reinstall TMIF 332c. Test module 330c can then cancel the reinstall and may attempt to reinstall TMIF 332c at a later time. --